The Journal of Open Source Software

[illegible]

Mathieu Besançon[illegible]

[illegible]

DOI: [illegible]

Software

- [illegible]
- [illegible]
- [illegible]

Submitted: [illegible]
Published: [illegible]

Summary

[illegible]

[illegible]

Bilevel optimization

[illegible]

```
min_x F(x,y)
s.t.  G_k(x,y) <= 0 for k in {1..mu}
      y in arg min_y {f(x,y)
                s.t.
                g_i(x,y) <= 0 for i in {1..ml}
               }
```

[illegible] $f(x, y)$ [illegible] $g_i(x, \cdot)$ [illegible]

[illegible] *BilevelOptimization.jl* [illegible]

[illegible]

```
min_x cx^T x + cy^T y
s.t. G x + H y <= q
     x >= 0
     x[j] integer for j in Jx
     y in arg min_y {d^T y + x^T F y
           s.t. A x + B y <= b
                y >= 0
                }
```

¦ \0 _jXe0]e0®0e }0L«H§nj nT §\0 nq§_h_ §H ®0} _nj nT §_ q}n@e0h _ I

```
min_{x,y,lambda} cx^T x + cy^T y
s.t. G x + H y <= q
     A x + B y <= b
     x, y >= 0
     x[j] integer for j in Jx
     d + F^T x + B^T * lambda = 0
     lambda[i] * (b - A x - B y)[i] = 0 for i in {1..ml}
```

¦ \0 e9 § 0x«9§nj _ 9 Hnhqe0h0j§9}_§´ Hnj §}9_j§J Hn}}0 qnjL_jX §n §\0 T9H§ §\9§ 9§ e09 § nj0 nT (λ_i, s_i) \9 §n @0 0x«9e §n ¶0}nJ ° \0}0 s_i _ §\0 e9Hc ®9}_9@e0 9 nH_9§0L °_§\ Hnj §}9_j§ iu ¦ _ jnj]Hnj®0²J jnj]e_j09} Hnj §}9_j§ H9jjn§ @0 §9HceOL 0¹H_0j§e´ @´ Hnhhnj nq§_h_¶9§_nj ne®0} 9jL j00L §n @0 }0]Tn}h«e9§0Lu ¦ \0 §°n Hnhhnj 9qq}n9H\0 9}0 e_j09}_¶9§_nj « _jX 9 @_j9}´ ®9}_9@e0 9jL ō@_X] gŸ q}_h9e 9jL L«9e «qq0} @n«jL 9jL ~q0H_9e m}L0}0L ~0§ nT §´q0 o r~m~os rp_j0L9 : gn}9e0 J ©·oksu 8 q0H_9e n}L0}0L 0§ o _ 9 X}n«q nT §°n n} hn}0 ®9}_9@e0 J nT °_H\ 9§ hn § nj0 H9j @0 jnj]¶0}nu g_²0L]^j§0X0} d_j09} ne®0} « 0 §_ _jTn}h9§_nj Tn} @}9jH_jX L_}0H§e´ nj §\0 §°n ®9}_9@e0 u ^j §\0 H9 0 nT §\0 @_e0®0e q}n@e0h q}0 0j§0L 9@n®0J §\0 0§ Hnj§9_j §\0 e9Hc ®9}_9@e0 9jL L«9e ®9}_9@e0 9 nH_9§0L °_§\ 09H\ en°0}]e0®0e Hnj §}9_j§J Tn}H_jX 9§ e09 § nj0 nT §\0h §n ¶0}nu

Types and methods for bilevel optimization

¦ \0 L9§9 Tn} 9 @_e0®0e e_j09} q}n@e0h 9}0 §n}0L _j §\0 `BilevelLP` §}«H§«}0J °_§\ §\0 9h0 jn§9§_nj 9 « 0L 9@n®0u 8 `JuMP.Model` H9j @0 @«_e§ T}nh H}9§H\ T}nh §\0 0 L9§9 n} q9 0L nj §n §\0 `build_blp_model` T«jH§_nj °_H\ 9LL §\0 en°0}]e0®0e T09 _@_e_§´ 9jL nq§_h9e_§´ Hnj §}9_j§ u ¦ _ T«jH§_nj «qqn}§ L_¸0}0j§ _Xj9§«}0 §\}n«X\ h«e§_qe0 L_ q9§H\u ¦n @«_eL §\0 hnL0e T}nh H}9§H\ « _jX §\0 L9§9 T}nh `BilevelLP`J §\0 T«jH§_nj _ H9ee0L 9 Tneen° I

```
build_blp_model(bp::BilevelLP, solver; [comp_method])
```

^§ }0§«}j 9 §«qe0 `(m, x, y, lambda, s)` °_§\I

- ą `m`I §\0 `«gp hnL0eJ
- ą `x`I §\0 «qq0}]e0®0e ®0H§n} nT ®9}_9@e0 J
- ą `y`I §\0 en°0}]e0®0e ®0H§n} nT ®9}_9@e0 J
- ą `lambda`I §\0 en°0}]e0®0e L«9e ®9}_9@e0 J 9jL
- ą `s`I §\0 en°0}]e0®0e e9Hc ®9}_9@e0 u

m§\0} _Xj9§«}0 9een° « 0} §n hnL_T´ 9j 9e}09L´]0²_ §_jX `«gp hnL0e °_§\n«§ §n}_jX 0®0}´ Hnj §}9_j§ n} ®9}_9@e0 _j9 `BilevelLP` §}«H§«}0u

All signatures of the `build_blp_model` function include a keyword argument `comp_method` for the choice of method to tackle complementarity constraints. The two methods mentioned in the previous section are represented as types, `SOS1Complementarity` and `BoundComplementarity`. For the second option, primal and dual bounds can either be scalars or vectors to use bounds adapted to each constraint. Some problem-specifc methods are often used in the literature to handle complementarity constraints in an e f cient way. Users of the package can defne a new type, optionally sub-typed from `ComplementarityMethod`, and defne the following function:

```
add_complementarity_constraint(m, cm::CM, s, lambda, y, sigma)
```

where CM is the complementarity constraint type.

Application to toll-setting problems

The toll-setting problem is a class of bilevel optimization where the two levels of decision are taken on a graph (Brotcorne, Labbé, Marcotte, & Savard, 2001). It belongs to the more general framework of network pricing problems with applications in road management (Harks, Schröder, & Vermeulen, 2019) or telecommunication network reliability (Hayrapetyan, Tardos, & Wexler, 2007).

In this problem, the upper level decides on a toll to apply on some arcs of a directed graph. Each arc has an initial cost and a cost resulting from an upper-level decision. The lower-level problem then consists in fnding the minimum-cost fow from a source to a sink with a minimum circulating fow. This problem can be entirely modeled using the framework presented above, using a composite datatype defned in the package for holding all required data, and allowing users to bypass the re-formulation of the model from its algebraic JuMP form to a standard form. Users can describe their problem using the `BilevelFlowProblem` struct containing:

- The initial matrix of arc costs `init_cost`,
- A boolean matrix indicating which edges are taxable `taxable_edges`,
- The capacity matrix `capacities`,
- The diferent levels of tax that can be applied to each arc `tax_options`, and
- The minimum amount of fow that the follower needs to pass from source to sink `minflow`

Building the `JuMP.Model` is done similarly to the generic bilevel problem, using the following signature:

```
build_blp_model(bfp::BilevelFlowProblem, solver; [comp_method])
```

More general problem formulations

Even though BilevelOptimization.jl is designed for linear-linear bilevel problems which can be described by the `BilevelLP` type, the API allows users to bypass the upper-level problem specifcation. They can provide a pre-built `JuMP.Model` with the upper-level objective and constraints already set, for instance for quadratic or conic upper level formulations. The only requirement is that the solver must support both the type of constraints specifed in the model and in the `comp_method`. This fexibility allows users to leverage some recent advances on mixed-integer convex optimization and solvers tackling these problems (Lubin, Yamangil, Bent, & Vielma, 2016). As of the current state of BilevelOptimization.jl, the only restricted part of the model is the linear-quadratic lower-level, which is required to exploit Karush-Kuhn-Tucker conditions.

References

Brotcorne, L., Labbé, M., Marcotte, P., & Savard, G. (2001). A bilevel model for toll optimization on a multicommodity transportation network. *Transportation science*, *35*(4), 345–358. doi:10.1287/trsc.35.4.345.10433

Dempe, S. (2018). *Bilevel optimization: Theory, algorithms and applications*. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik.

Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, *59*(2), 295–320. doi:10.1137/15m1020575

Harks, T., Schröder, M., & Vermeulen, D. (2019). Toll caps in privatized road networks. *European Journal of Operational Research*. doi:10.1016/j.ejor.2019.01.059

Hayrapetyan, A., Tardos, É., & Wexler, T. (2007). A network pricing game for selfsh trafc. *Distributed Computing*, *19*(4), 255–266. doi:10.1007/s00446-006-0020-y

Lubin, M., Yamangil, E., Bent, R., & Vielma, J. P. (2016). Extended formulations in mixed-integer convex programming. In Q. Louveaux & M. Skutella (Eds.), *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings* (pp. 102–113). Cham: Springer International Publishing. doi:10.1007/978-3-319-33461-5_9

Pineda, S., & Morales, J. M. (2019). Solving linear bilevel problems using big-ms: Not all that glitters is gold. *IEEE Transactions on Power Systems*, 1–1. doi:10.1109/TPWRS.2019.2892607